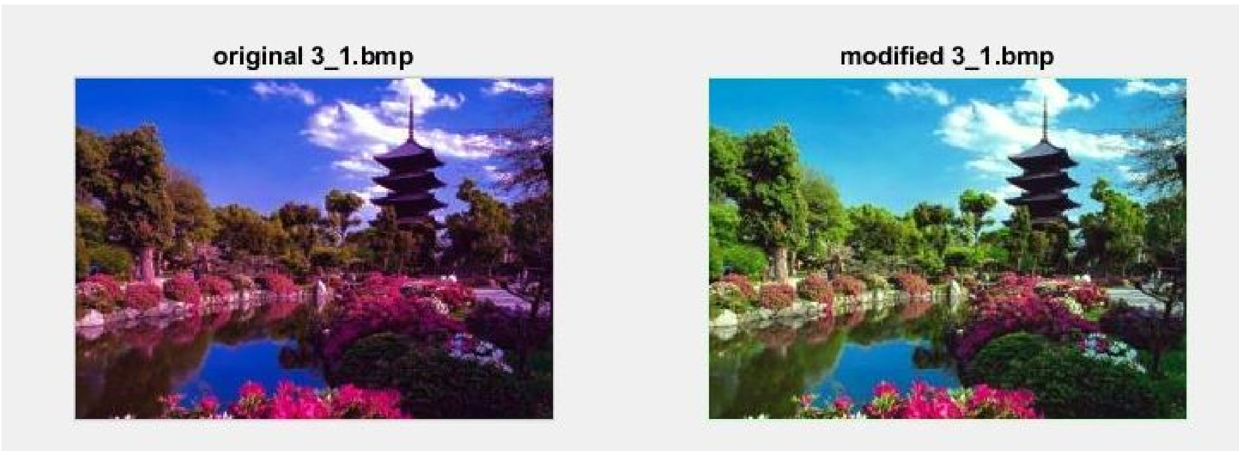
EE 440 Homework #3

(Luke Jiang, 1560831)

1)

By comparing the original image and the expected output, I found that the original image is heavy in magenta. Therefore, I used Power-Law transform with gamma = 2.5 and c = 1 to make the image look more natural.



Source Code:

```
% 3.1. Remove extra magenta content from 3_1.bmp to make the image look
% more natural.
% -read image 3_1
img = double(imread('3_1.bmp'));
figure(1);
subplot(1, 2, 1);
    imshow(img / 255);
    title('original 3\_1.bmp');
% -M255 is used for conversion between RGB and CMY
M255 = 255 * ones(202, 282, 3);
% -get the CMY image
img_cmy = M255 - img;
C = img_cmy(:,:,1);
M = img_cmy(:,:,2);
Y = img_cmy(:,:,3);
% -the input image is heavy in magneta, use Power-Law Transform with
% -gamma = 2.5 to create a weaker version of the magneta band
M_weaker = ((M/255).^ 2.5)*255;
% -replace the original magneta band with the weakened version
img_weaker = img_cmy;
img_weaker(:,:,2) = M_weaker;
% -convert back to RGB and display
img_weaker = M255 - img_weaker;
subplot(1, 2, 2);
    imshow(img_weaker / 255);
    title('modified 3\_1.bmp');
```

2)

-a. Processing 3_2.jpg:

-linear stretching:

From histogram for V band of original 3_2.jpg, I found that the majority of pixels lies in the range [0, 0.6]. Therefore, I multiplied V with coefficient 1/0.6 = 5/2 to produce V_str whose histogram is the 2nd picture in 2.2.

-histogram equalization:

I used library function imhist to divide [0, 1] interval of possible intensities into 256 parts (step = 0.0039), and used the algorithm covered in class to calculate the resultant intensities:

$$h(v) = round(cdf(v) / (M*N) * (L-1))$$

Where M*N is the total number of pixels of the image and (L-1) is the range of intensities of the image. For 3_2.jpg, M*N = 68400 and (L-1) = 1.

-histogram specification:

I used library function normpdf(0:0.0039:1, 1, 0.39) to generate a normally distributed sequence of 256 entries centered at 1 with standard deviation = 0.39.

-results:

All methods made the image looks brighter, but the picture generated histogram specification looks most natural.

-b Processing 3_3.jpg:

-linear stretching:

From histogram for V band of original 3_3.jpg, I found that the majority of pixels lies also in the range [0, 0.6]. Therefore, I used the same procedure as with 3_2.jpg.

-histogram equalization & histogram specification:

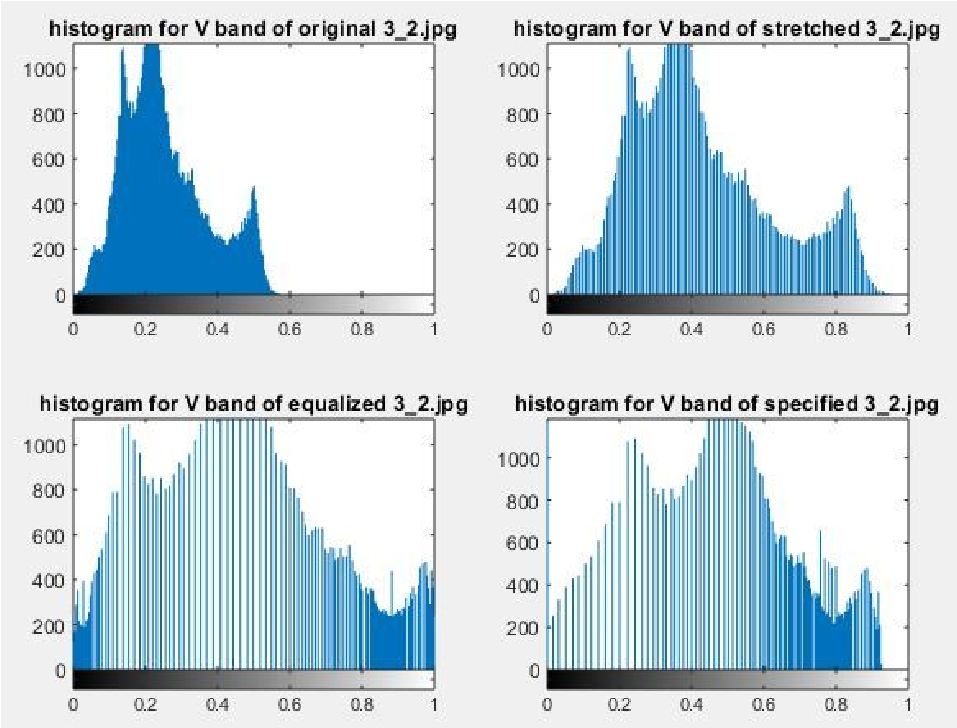
Same as 3_2.jpg

-results:

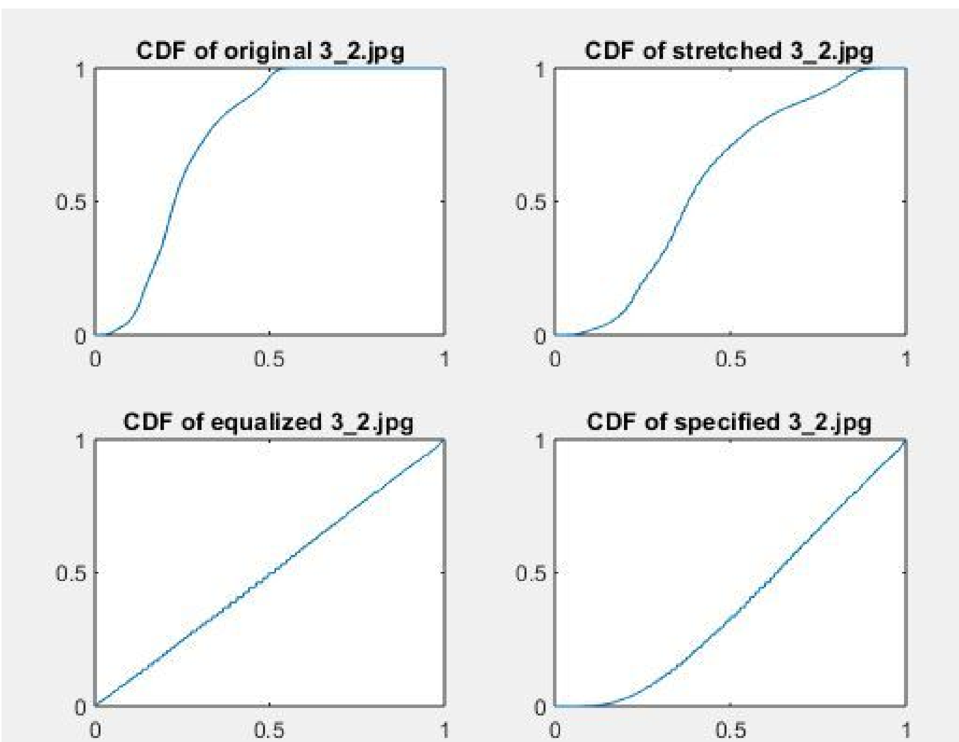
The output of histogram equalization and histogram specification are very similar, and both of them are far inferior to the result of linear stretching in terms of image quality.


original 3_2.jpg
linear stretched 3_2.jpg
Histogram equalized 3_2.jpg
Histogram specified 3_2.jpg

(2.1)


histogram for V band of original 3_2.jpg
histogram for V band of stretched 3_2.jpg
histogram for V band of equalized 3_2.jpg
histogram for V band of specified 3_2.jpg

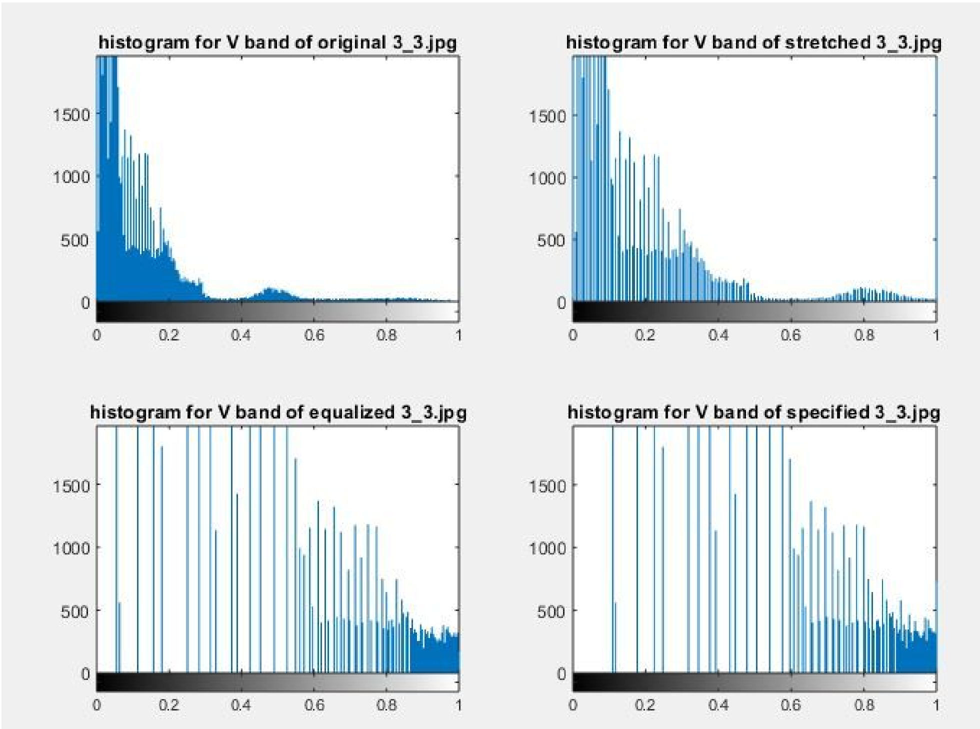
(2.2)


CDF of original 3_2.jpg
CDF of stretched 3_2.jpg
CDF of equalized 3_2.jpg
CDF of specified 3_2.jpg

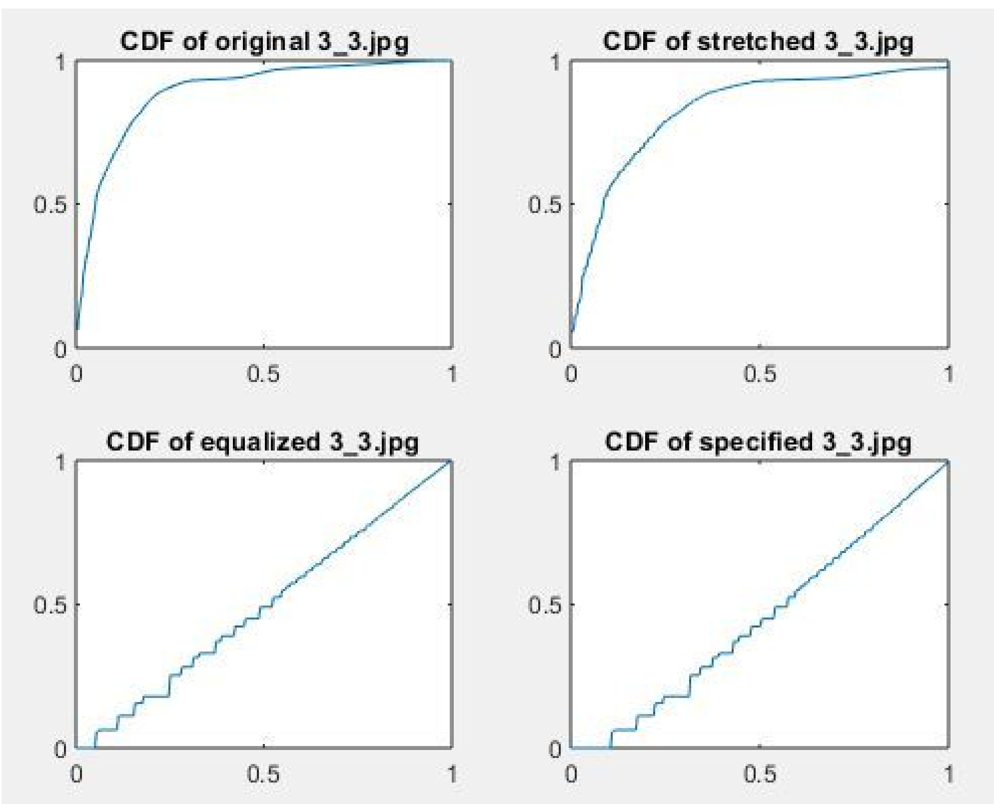
(2.3)


original 3_3.jpg
linear stretched 3_3.jpg
Histogram equalized 3_3.jpg
Histogram specified 3_3.jpg

(2.4)


histogram for V band of original 3_3.jpg
histogram for V band of stretched 3_3.jpg
histogram for V band of equalized 3_3.jpg
histogram for V band of specified 3_3.jpg

(2.5)


CDF of original 3_3.jpg
CDF of stretched 3_3.jpg
CDF of equalized 3_3.jpg
CDF of specified 3_3.jpg

(2.6)

Source Code:

A:

```
% -parameters for drawing
IM2_RES = 2;
IM2_HIST = 3;
IM2_CDF = 4;
% -read and display the orignal image
im2 = imread('3_2.jpg');
figure(IM2_RES); subplot(2, 2, 1);
    imshow(im2);
    title('original 3\_2.jpg');
% -convert to HSV and get the histogram of the V band
im2_hsv = rgb2hsv(im2);
V2 = im2_hsv(:,:,3);
V2_X = size(V2, 1);
V2_Y = size(V2, 2);
figure(IM2_HIST); subplot(2, 2, 1);
    imhist(V2);
    title('histogram for V band of original 3\_2.jpg');
% -get CDF and plot it
[binLocations, cdf2] = getCDF(V2, IM2_CDF, 2, 2, 1, 'CDF of original 3\_2.jpg');

% a) Linear Stretching:
% -from hist see that V mostly falls in [0, 0.6] => 5/3
% -produce stretched V as V_str and display its histogram
V2_str = V2 * 5/3;
figure(IM2_HIST); subplot(2, 2, 2);
    imhist(V2_str);
    title('histogram for V band of stretched 3\_2.jpg');
% -get equalized CDF and display
getCDF(V2_str, IM2_CDF, 2, 2, 2, 'CDF of stretched 3\_2.jpg');
% -get result image, display and save
VReplaceDisplaySave(im2_hsv, V2_str, IM2_RES, 2, 2, 2, ...
'linear stretched 3\_2.jpg', 'im2_stretched.jpg');
```

```
% b) Histogram Equalization:
% -calculate equalized V band
V2_eq = cdf2(round(V2 / 0.0039) + 1);
% -display the equalized histogram
figure(IM2_HIST); subplot(2, 2, 3);
    imhist(V2_eq);
    title('histogram for V band of equalized 3\_2.jpg');
% -get equalized CDF and display
getCDF(V2_eq, IM2_CDF, 2, 2, 3, 'CDF of equalized 3\_2.jpg');
% -get result image, display and save
VReplaceDisplaySave(im2_hsv, V2_eq, IM2_RES, 2, 2, 3, ...
'Histogram equalized 3\_2.jpg', 'im2_equalized.jpg');

% c) Histogram Specification:
% -use a specified normal distribution (sp) as the target histogram
sp2 = normpdf(0:0.0039:1, 1, 0.39)';
sp2 = sp2(1:end-1);
cdf_sp2 = cumsum(cdf2);
cdf_sp2 = cdf_sp2 / max(cdf_sp2);
% -for each pixel, get the index of the sp value that is smaller and
% most close to the cdf of this pixel.
V2_sp = zeros(V2_X, V2_Y);
for i = 1:V2_X
    for j = 1:V2_Y
        cdf_ij = cdf2(round(V2(i, j) / 0.0039) + 1);
        index = 1;
        while (cdf_sp2(index) < cdf_ij)
            index = index + 1;
        end
        V2_sp(i, j) = binLocations(index);
    end
end
```

```
% -display the specified histogram
figure(IM2_HIST); subplot(2, 2, 4);
    imhist(V2_sp);
    title('histogram for V band of specified 3\_2.jpg');
% -get specified CDF and display
getCDF(V2_sp, IM2_CDF, 2, 2, 4, 'CDF of specified 3\_2.jpg');
% -get result image, display and save
VReplaceDisplaySave(im2_hsv, V2_sp, IM2_RES, 2, 2, 4, ...
'Histogram specified 3\_2.jpg', 'im2_specified.jpg');
```

B:

```
% -parameters for drawing
IM3_RES = 5;
IM3_HIST = 6;
IM3_CDF = 7;
% -read and display the orignal image
im3 = imread('3_3.jpg');
figure(IM3_RES); subplot(2, 2, 1);
    imshow(im3);
    title('original 3\_3.jpg');
% -convert to HSV and get the histogram of the V band
im3_hsv = rgb2hsv(im3);
V3 = im3_hsv(:,:,3);
V3_X = size(V3, 1);
V3_Y = size(V3, 2);
figure(IM3_HIST); subplot(2, 2, 1);
    imhist(V3);
    title('histogram for V band of original 3\_3.jpg');
% get CDF and plot it:
[binLocations3, cdf3] = getCDF(V3, IM3_CDF, 2, 2, 1, 'CDF of original 3\_3.jpg');
cdf3 = [cdf3; 1];  % fix off-by-one error
binLocations3 = [binLocations3; 1];  % fix off-by-one error
% a) Linear Stretching:
% -from hist see that V mostly falls in [0, 0.6] => 5/3
% -produce stretched V as V_str and display its histogram
V3_str = V3 * 5/3;
figure(IM3_HIST); subplot(2, 2, 2);
    imhist(V3_str);
    title('histogram for V band of stretched 3\_3.jpg');
% -get equalized CDF and display
getCDF(V3_str, IM3_CDF, 2, 2, 2, 'CDF of stretched 3\_3.jpg');
% -get result image, display and save
```

```
VReplaceDisplaySave(im3_hsv, V3_str, IM3_RES, 2, 2, 2, ...
'linear stretched 3\_3.jpg', 'im3_stretched.jpg');

% b) Histogram Equalization:
% -calculate equalized V band
V3_eq = cdf3(round(V3 / 0.0039) + 1);
% -display the equalized histogram
figure(IM3_HIST); subplot(2, 2, 3);
    imhist(V3_eq);
    title('histogram for V band of equalized 3\_3.jpg');
% -get equalized CDF and display
getCDF(V3_eq, IM3_CDF, 2, 2, 3, 'CDF of equalized 3\_3.jpg');
% -get result image, display and save
VReplaceDisplaySave(im3_hsv, V3_eq, IM3_RES, 2, 2, 3, ...
'Histogram equalized 3\_3.jpg', 'im3_equalized.jpg');

% c) Histogram Specification:
% -use a specified normal distribution (sp) as the target histogram
sp3 = normpdf(0:0.0039:1, 1, 0.39)';
sp3 = sp3(1:end-1);
cdf_sp3 = cumsum(cdf3);
cdf_sp3 = cdf_sp3 / max(cdf_sp3);
% -for each pixel, get the index of the sp value that is smaller and
% most close to the cdf of this pixel.
V3_sp = zeros(240, 320);
for i = 1:240
    for j = 1:320
        cdf_ij = cdf3(round(V3(i, j) / 0.0039) + 1);
        index = 1;
        while (cdf_sp3(index) < cdf_ij)
            index = index + 1;
        end
        V3_sp(i, j) = binLocations3(index);
```

```
end
% -display the specified histogram
figure(IM3_HIST); subplot(2, 2, 4);
    imhist(V3_sp);
    title('histogram for V band of specified 3\_3.jpg');
% -get specified CDF and display
getCDF(V3_sp, IM3_CDF, 2, 2, 4, 'CDF of specified 3\_3.jpg');
% -get result image, display and save
VReplaceDisplaySave(im3_hsv, V3_sp, IM3_RES, 2, 2, 4, ...
'Histogram specified 3\_3.jpg', 'im3_specified.jpg');
```

Helper functions:
To make the code cleaner and more organized, I wrote two customary functions to encapsulate repetitive logics.
-getCDF

```
% for a given image, plot the CDF of the pixels in the image and return
% the CDF

% @param V: input image
% @param fignum, a, b, c: which figure, (a, b, c) are subplot inputs
% @param ttl: the title of the plot
% @return binLocations: binLocations returned by imhist
% @return cdf: computed CDF

function [binLocations, cdf] = getCDF(V, fignum, a, b, c, ttl)
    [counts_, binLocations_] = imhist(V);
    counts = counts_;
    binLocations = binLocations_;
    % -get normalized CDF
    cdf = cumsum(counts) / (size(V, 1) * size(V, 2));
    % -plot the CDF
    figure(fignum); subplot(a, b, c);
    plot(binLocations, cdf);
    title(ttl);
end
```

-VPeplaceDisplaySave:

```
% given a HSV file and a V band, replace the original V band with the input
% V band, convert the image back to RGB and display, save, and return the
% image

% @param hsv: input image in HSV format
% @param V: input V band to replace V band in hsv
% @param fignum, a, b, c: which figure, (a, b, c) are subplot inputs
% @param ttl: the title of the plot
% @param filename: the filename of the saved image
% @return res: the resultant RGB image
function res = VReplaceDisplaySave(hsv, V, fignum, a, b, c, ttl, filename)
    res = hsv;
    res(:,:,3) = V;
    res = hsv2rgb(res);
    figure(fignum); subplot(a, b, c);
    imshow(res);
    title(ttl);
    imwrite(res, filename);
end
```